Advanced Data Science & Python Stock Analysis

Final Project

Part 2 - Common Financial Analysis

Creator: Wendy(Aobo) Liu

In [1]:
```
!pip install intrinio_sdk
```

```
Requirement already satisfied: intrinio_sdk in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (5.5.0)
Requirement already satisfied: six>=1.10 in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (from intrinio_sdk)
(1.11.0)
Requirement already satisfied: urllib3>=1.15 in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (from intrinio_sd
k) (1.23)
Requirement already satisfied: certifi in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (from intrinio_sdk) (20
18.10.15)
Requirement already satisfied: python-dateutil in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (from intrinio_
sdk) (2.8.1)
WARNING: You are using pip version 19.3.1; however, version 20.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [3]:
```
import numpy as np
import pandas as pd
import intrinio_sdk
import configparser as cp
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

2.1. Five companies in the selected industry:Data Processing and Outsourced Services.

Download of 120 trading days of data using the Intrinio API.

In [5]:
```
sp_df = pd.read_csv('../data/SP1500.csv',index_col=0)
sp_df[sp_df['industry']=='Data Processing and Outsourced Services'].sort_values('marketcap_mm').tail()
```

Out[5]:

	company	ticker	price_close	pct_price_change_lastday	pct_price_change_30day	pct_price_change_ytd	pct_price_change_12_month	P/E*†	P/BV*†	marke
522	Fiserv, Inc.	FISV	101.54	1.29	17.81	-12.19	17.06	58.50x	2.09x	
507	Fidelity National Information Services, Inc.	FIS	128.58	-0.55	11.60	-7.56	9.19	195.35x	1.60x	
1014	PayPal Holdings, Inc.	PYPL	123.66	2.53	33.85	14.32	10.45	58.17x	8.59x	1
835	Mastercard Incorporated	MA	269.26	0.19	13.60	-9.82	8.59	34.26x	50.08x	2
1435	Visa Inc.	V	176.15	0.33	16.00	-6.25	8.71	32.10x	12.91x	3

In [6]:
```
cfg = cp.ConfigParser()
cfg.read('../resources/credentials.cfg')
```

Out[6]:
```
['../resources/credentials.cfg']
```

In [7]:
```
API_KEY = cfg['intrinio']['app_key']

intrinio_sdk.ApiClient().configuration.api_key['api_key'] = API_KEY

security_api = intrinio_sdk.SecurityApi()
```

In [8]:
```
# ~120 Trading Days
len(pd.bdate_range('2019-11-15','2020-04-30'))
```

Out[8]:
```
120
```

In [10]:
```
tickers = sp_df[sp_df['industry']=='Data Processing and Outsourced Services'].ticker.values
start_date = '2019-11-15'
end_date = '2020-04-30'
frequency = 'daily'
```

Making multiple request to Intrinio API

In [11]:
```
dfs = []

for ticker in tickers:
    next_page = ''
    response = security_api.get_security_stock_prices(ticker,
                                                       start_date = start_date,
                                                       end_date = end_date)
    df = [p.to_dict() for p in response.stock_prices]
    next_page = response.next_page
    if next_page != None:
        response = security_api.get_security_stock_prices(ticker,
                                                           start_date = start_date,
                                                           end_date = end_date,
                                                           next_page = next_page)
        df.extend(p.to_dict() for p in response.stock_prices)
    df = pd.DataFrame.from_dict(df)
    df['secid'] = ticker
    dfs.append(df)
```

In [12]:
```
data_df = pd.concat(dfs)
data_df.index = pd.DatetimeIndex(data_df['date'])
data_df = data_df.drop('date', axis=1)
data_df.index.name = None

#SORT DATETIME INDEX
data_df = data_df.sort_index()
data_df.shape
```

Out[12]:
```
(2622, 13)
```

In [13]:
```
data_df.to_csv("../data/data_df.csv")
```

In [14]:
```
data_df.head()
```

Out[14]:

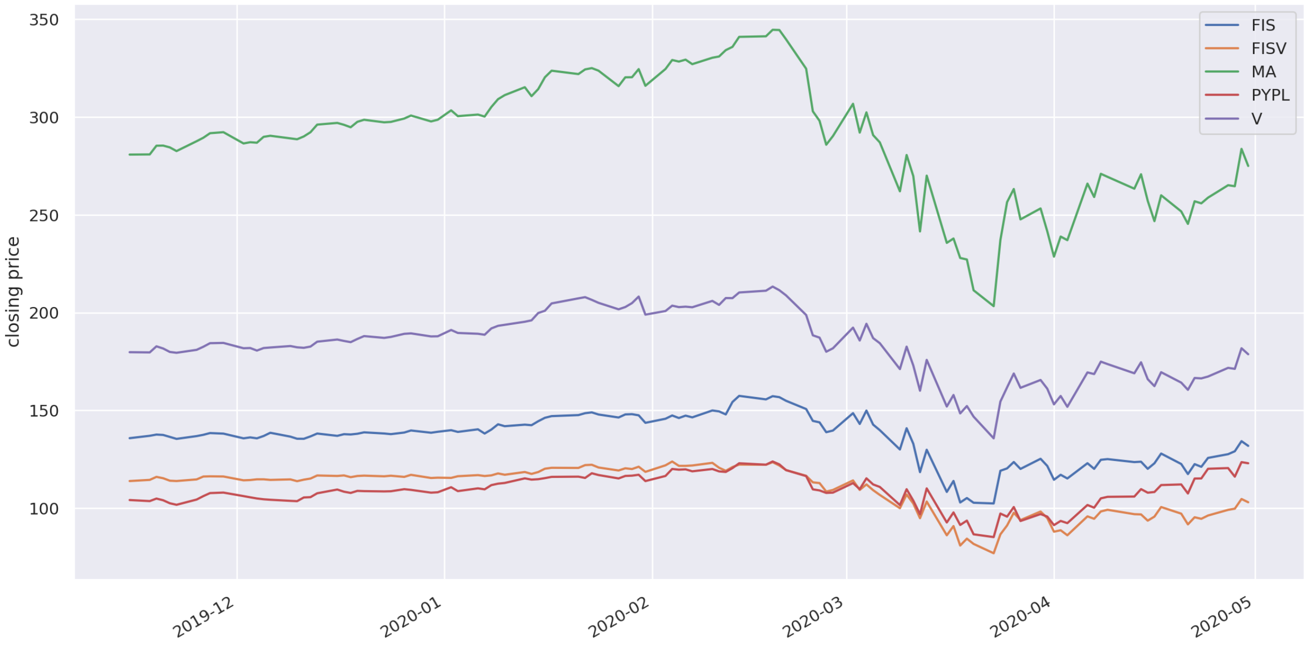
	adj_close	adj_high	adj_low	adj_open	adj_volume	close	frequency	high	intraperiod	low	open	volume	secid
2019-11-15	200.550000	200.830000	195.950000	197.420000	269874.0	200.55	daily	200.83	False	195.95	197.42	269874.0	WEX
2019-11-15	35.170000	35.740000	35.070000	35.740000	705534.0	35.17	daily	35.74	False	35.07	35.74	705534.0	SYKE
2019-11-15	40.600000	40.750000	39.940000	40.190000	362497.0	40.60	daily	40.75	False	39.94	40.19	362497.0	CATM
2019-11-15	102.824054	106.223361	102.217745	106.223361	1486284.0	103.45	daily	106.87	False	102.84	106.87	1486284.0	ADS
2019-11-15	179.510016	180.418700	178.821014	179.769640	7809545.0	179.77	daily	180.68	False	179.08	180.03	7809545.0	V

2.2 Plot the closing price of each security on the same chart.

In [15]:
```
companies = sp_df[sp_df['industry']=='Data Processing and Outsourced Services'].sort_values('marketcap_mm').tail().ticker.v
data_df = data_df[[i in companies for i in data_df.secid]]
data_df.secid.value_counts()
```

Out[15]:
```
V       114
MA      114
FISV    114
FIS     114
PYPL    114
Name: secid, dtype: int64
```

In [26]:
```
plt.figure(figsize=(15,8), dpi=200)
data_df.groupby('secid').close.plot()
plt.legend()
plt.ylabel('closing price')
plt.savefig("../graph/closing_price.jpg")
```



2.3 Calculate and plot the returns and log returns for each security.

In [20]:
```
returns = {data_df.secid.unique()[i]:pd.DataFrame([*data_df.groupby('secid')][i][1].adj_close.pct_change()) for i in range(
return_df = pd.concat([*returns.values()],axis=1,ignore_index=True)
return_df.columns = returns.keys()
logreturn_df = np.log(return_df + 1)
```

In [21]:
```
return_df.head()
```

Out[21]:

	V	PYPL	MA	FISV	FIS
2019-11-15	NaN	NaN	NaN	NaN	NaN
2019-11-18	0.008982	0.005180	0.000285	-0.005182	-0.000612
2019-11-19	0.004524	0.013450	0.015844	0.012445	0.017310
2019-11-20	-0.001380	-0.005946	0.000245	-0.008194	-0.006073
2019-11-21	-0.007274	-0.011010	-0.003259	-0.014795	-0.009743

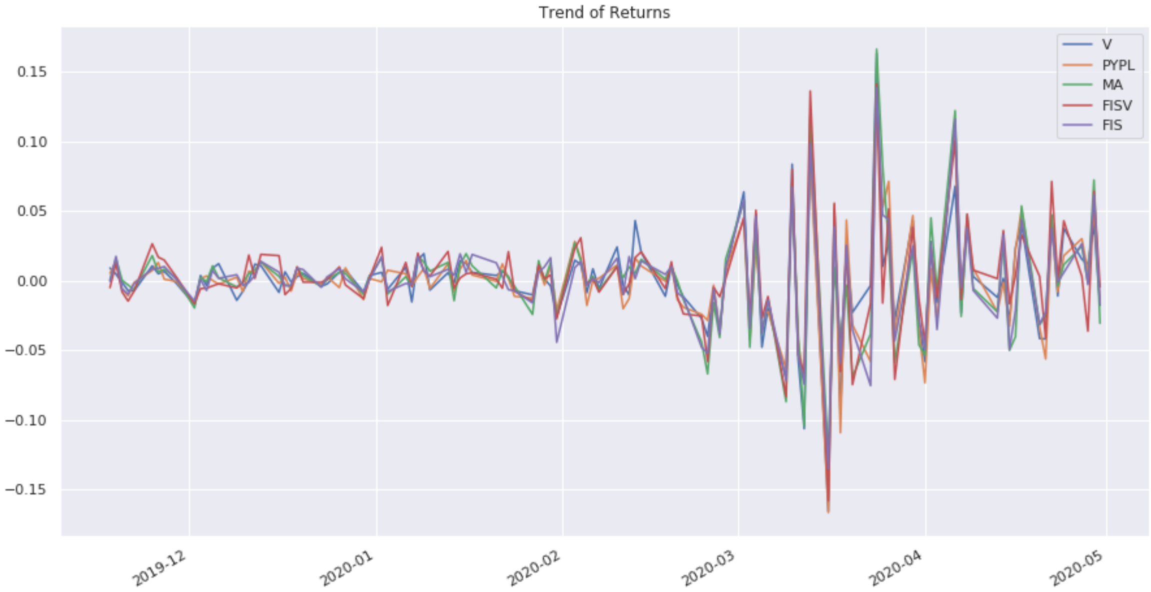
In [22]:
```
logreturn_df.head()
```

Out[22]:

	V	PYPL	MA	FISV	FIS
2019-11-15	NaN	NaN	NaN	NaN	NaN
2019-11-18	0.008942	0.005166	0.000285	-0.005196	-0.000612
2019-11-19	0.004514	0.013360	0.015720	0.012368	0.017162
2019-11-20	-0.001381	-0.005964	0.000245	-0.008228	-0.006092
2019-11-21	-0.007301	-0.011071	-0.003264	-0.014905	-0.009791

Trend of Returns

```
In [27]: return_df.plot(figsize=(15, 8))
         plt.title('Trend of Returns')
         plt.savefig("../graph/return_trends.jpg")
```

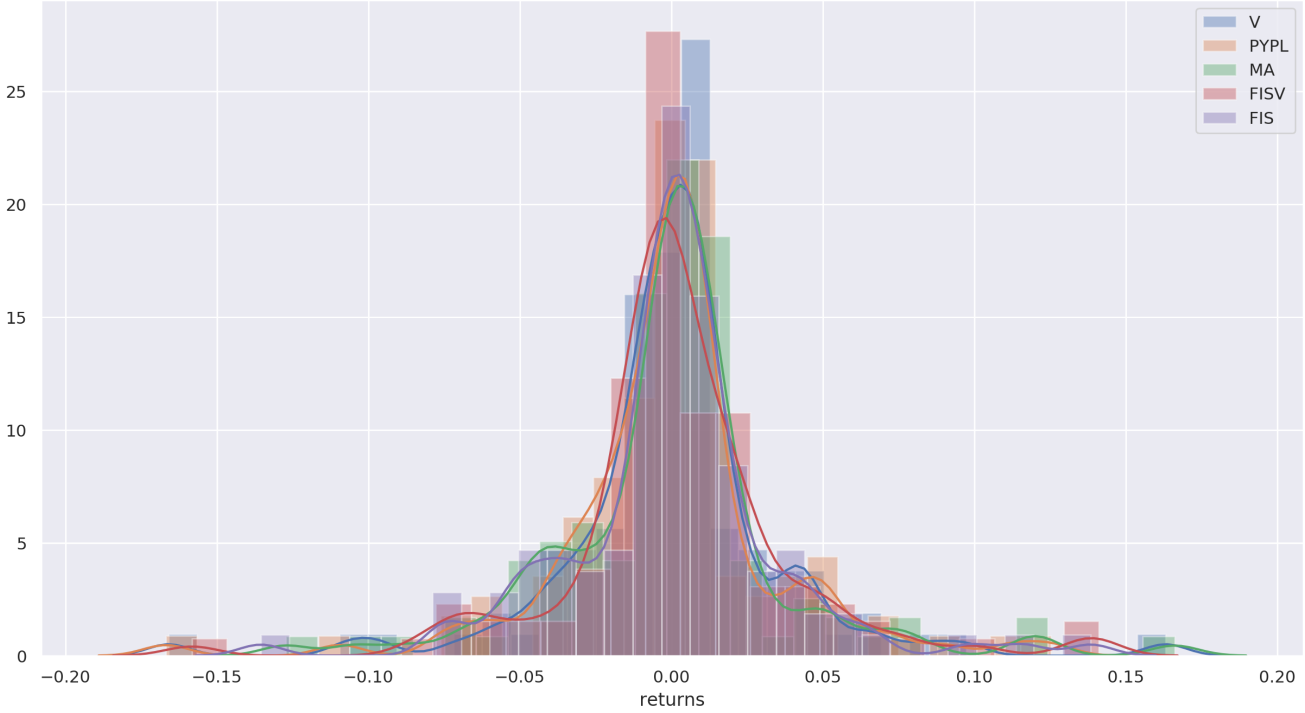


** Analysis **

From this chart, we can see that these five stock prices are higly correlated, having similar moving trends. This results from the similar risk factors those companies face and the same industry they belong.

Distribution of Returns

```
In [28]: plt.figure(figsize=(15,8), dpi=200)
         for i in return_df.columns:
             sns.distplot(return_df[i][1:])
         plt.legend(return_df.columns)
         plt.xlabel('returns')
         plt.savefig("../graph/distribution_of_return.jpg")
```



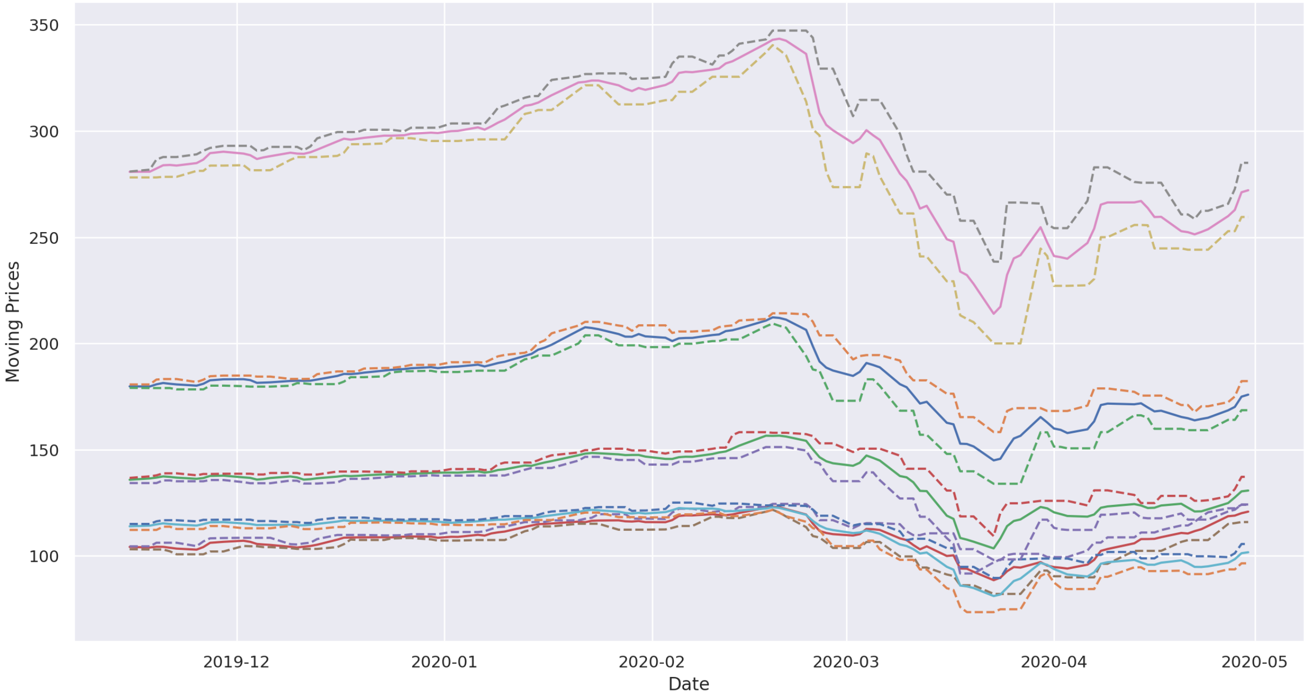
** Analysis **

Basicly, the daily returns of PYPL and V more concentrate around 0 and have lower volatility.

Moving/Window Statistics

In [29]:

```
plt.figure(figsize=(15,8), dpi=200)
for i in data_df.secid.unique():
    ma05_close = data_df[data_df['secid']==i]['close'].rolling('5D').mean()
    ma05_high = data_df[data_df['secid']==i]['high'].rolling('5D').max()
    ma05_low = data_df[data_df['secid']==i]['low'].rolling('5D').min()
    plt.plot(ma05_close)
    plt.plot(ma05_high,linestyle='--')
    plt.plot(ma05_low,linestyle='--')
plt.xlabel('Date')
plt.ylabel('Moving Prices')
plt.savefig("../graph/moving_Statistics.jpg")
```



In [ ]: